Syntax:

- `⊢` = Goal
- `∧` = And
- `∃` = There Exists
- `∀` = For All
- `have/def/theorem Name (Assumptions) : WhatYouWantToSay := ProofThatItsTrue`
- If you know the proof already, you don't need to say `by`. But if you have to build up the proof using a bunch of different facts, then start the proof with `by`.
  Let's review some tactics.
- When would you use `obtain`? If you have a hypothesis `H : ∃ thing, prop...` Then you can say `obtain ⟨thing, h_prop⟩ := H` and we will get the thing and the property.
- When would you use `have`? If you want to claim that something is true, e.g., `have Fact1 : IsRound Earth := by sorry`. (If we didn't name it `Fact1`, its default name would be `this`)
- When could you use `use`? Say you have `Jivin : Person` and the goal is to prove that there exists a person in the room

```
Jivin : Person
⊢ ∃ somebody : Person, IsInRoom somebody
```

Then if you write `use Jivin`, then the goal will change to:

```
⊢ IsInRoom Jivin
```

- When would you use `unfold`? If you don't remember what it means to be `IsWearingJeans`, say your goal was: `⊢ IsWearingJeans Navya` (And there was a definition:

```
def IsWearingJeans (somebody : Person) : Prop := IsWearingLongPants somebody
∧ PantsAreDenim somebody
```

If at this point, you write `unfold IsWearingJeans`, then the goal will become

```
⊢ IsWearingLongPants Navya ∧ PantsAreDenim Navya
```

- When would you write `constructor`? If your goal is to prove two things, then writing `constructor` gives you two different goals. For the previous line, `constructor` would result in:

```
⊢ IsWearingLongPants Navya
⊢ PantsAreDenim Navya
```

When we're in a situation where there are multiple goals, how do we focus in on just the one goal at a time? Type backslash-dot-space: `· ` will the tell the computer to not show the other goals.

- When would you write `convert`? Say you're trying to prove that

```
H : x + y = z + w
⊢ a + b = z + w
```

What will happen if I write `convert H`? If you don't restrict the `convert`, it will turn the goal into a pair of goals:

```
⊢ x = a
⊢ y = b
```

So maybe it's not true that `x = a`, so we don't want to convert this far. To restrict it, we say `convert H using 1`. Then the response will be:

```
⊢ x + y = a + b
```

(Or maybe `a + b = x + y`). Note: if `H : x = y`, then `H.symm : y = x`.

- When would you write `exact`? If your goal state is

```
H : IsWearingLongPants Navya
⊢ IsWearingLongPants Navya
```

then writing `exact H` will result in: No goals. (Done!)

- Suppose instead

```
H : ¬IsWearingLongPants Navya
⊢ IsWearingLongPants Navya
```

Gödel's incompleteness theorem.